Machine Learning for Modelling: *Supervised Learning*

Simone Bianco

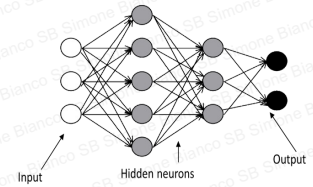
1

Convolutional Neural Networks (CNNs)

2

Multi-layer perceptron

- Neurons are divided in layers
  - of three different kinds: input, hidden, output
  - Each layer is connected to the next one (without cycles)
  - Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer.
  - Input data is used to set the input layer
  - The output layer produces the final outcome
  - For classification, one output neuron for each class
- Backpropagation is the most used training algorithm
  - Numerically fit the weights in order to produce output values similar to those in the training set



3

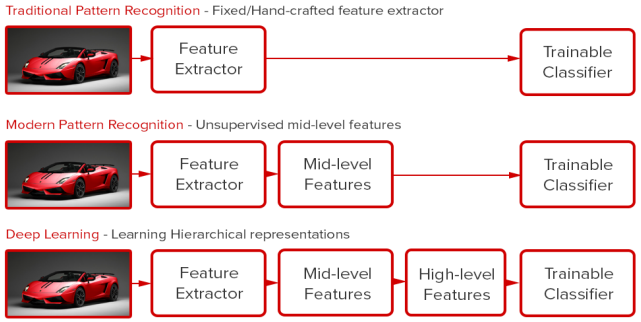
Multi-layer perceptron

Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer	Single hyperplane			
Two layers	Open or closed convex regions			
Three layers	Arbitrary (complexity limited by the number of nodes)			

FIGURE 12.23 Types of decision regions that can be formed by single- and multilayer feed-forward networks with one and two layers of hidden units and two inputs. (Lippman)

4

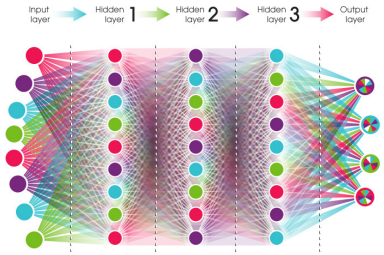
Hand-crafted vs learned features



5

What is Deep learning?

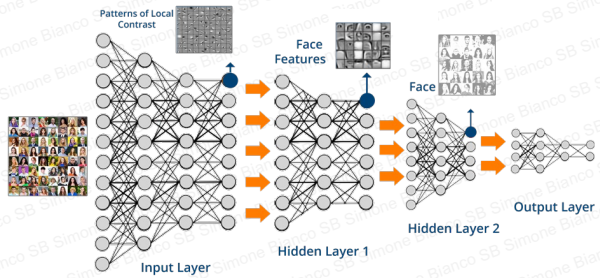
- Is a new area of Machine Learning research (ML), with the objective of moving ML closer to Artificial Intelligence
- It is about learning multiple levels (a hierarchy) of representations and abstractions



6

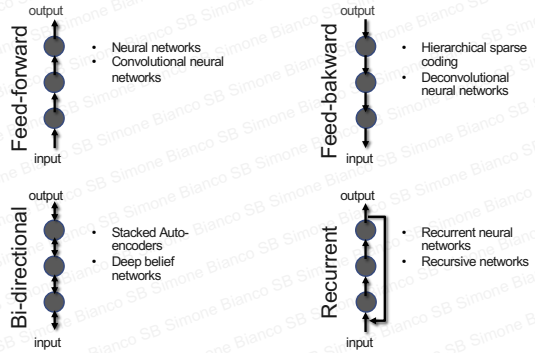
Feature hierarchy

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable non-linear feature transform
- Can share the lower-level representations for multiple tasks



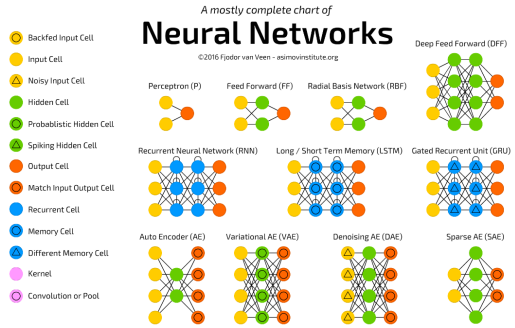
7

Main types of deep architectures



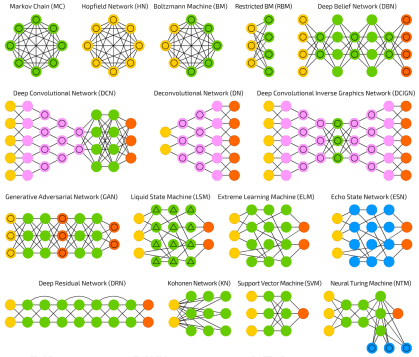
8

Many types of NN exist



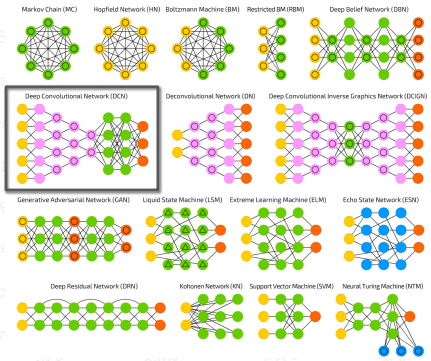
9

Many types of NN exist



10

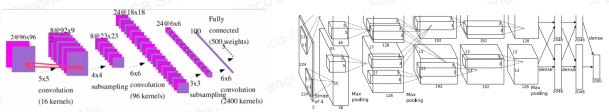
Many types of NN exist



11

Convolutional Neural Networks (CNNs)

- Convolutional Neural Networks are **biologically-inspired architectures** made up of neural networks stuff
- CNNs are the most used architectures in computer vision
- CNNs stacks multiple stages of feature extractors
- Higher stages compute more global, more invariant features

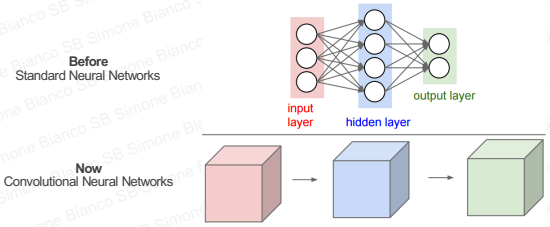


[1] Y. LeCun, et al., **Gradient-based learning applied to document recognition,** *Proceedings of the IEEE*, vol. 86, n. 11, pp. 2278-2324, *1998.*
[2] A. Krizhevsky et al., **Imagenet classification with deep convolutional neural networks**, *Advances in neural information processing systems, 2012.*

12

Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**



13

Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity

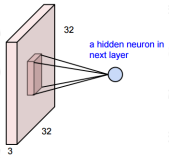


Image: 32x32x3 volume

Before: full connectivity 32x32x3 = 3072 weights

Now: one neuron will connect to, e.g. 5x5x3 patch and only have 5x5x3 = 75 weights

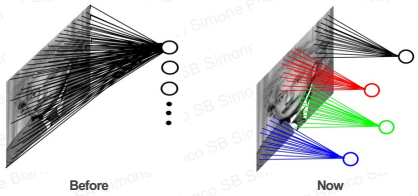
Note that connectivity is:

- Local in space (5x5 inside 32x32)
- Full in space (all 3 depth channels)

14

Convolutional Neural Networks (CNNs)

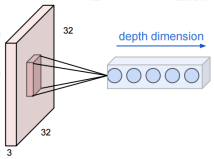
- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity



15

Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity



Before: hidden layer of 200 neurons

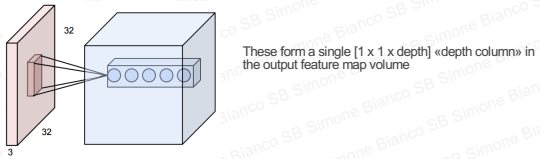
Now: output volume of depth 200

Multiple neurons all looking at the same region of the input volume, stacked along depth

16

Convolutional Neural Networks (CNNs)

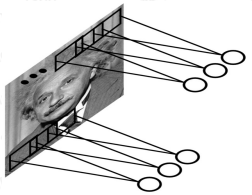
- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity



17

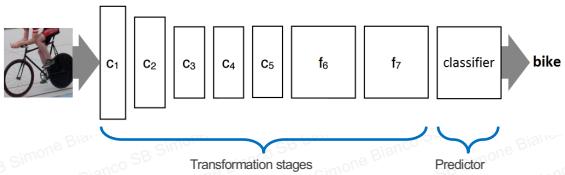
Convolutional Neural Networks (CNNs)

- All Neural Network weights arranged in **3 dimensions**
- Convolutional Neural Networks are still Neural Networks but:
  - Local connectivity
  - Weights sharing (i.e., weights in the layer are shared across spatial positions)



18

Overall CNN architecture



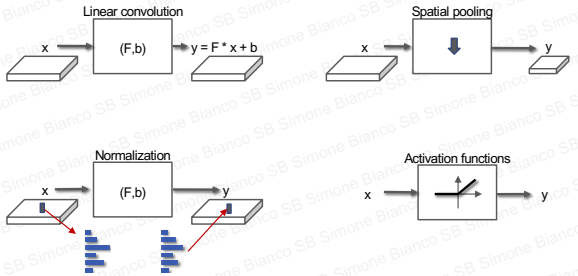
19

CNNs: key ideas

- Feed-forward feature extraction
- Supervised training of convolutional filters by back-propagating classification error
- With respect to traditional Neural Networks, CNNs have other special layers:
  - Spatial pooling
  - Local response normalization
- These layers are useful to:
  - Reduce computational burden
  - Increase invariance
  - Ease the optimization

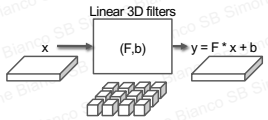
20

CNNs: components overview



21

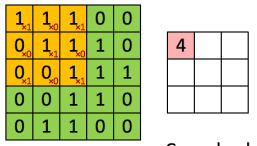
Linear convolution – Definition



- Linear
- Local
- Translation invariant
- Filter bank to form a richer representation of the data

- Input $x = H \times W \times K$ array
- Filter bank $\mathrm{F} = H' \times W' \times K \times Q$ array
- Output $y = (H - H' + 1) \times (W - W' + 1) \times Q$ array

$$y_{ijq} = y_q + \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} \sum_{k=1}^{K} x_{u+i,v+j,k} F_{u,v,k,q}$$

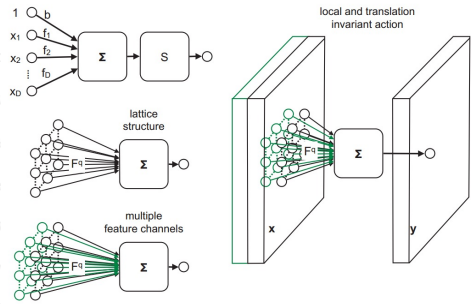


Image

Convolved Feature

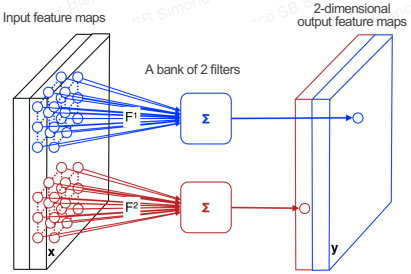
22

Linear convolution – As a neural network



23

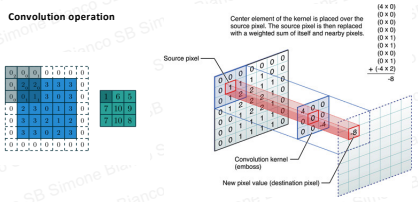
Linear convolution – As a neural network



24

24

Convolutional Layers



- Applies convolution to input feature maps
  - weights of the filters learned during training

➢ Parameters to set:
- Filter dimension
- Padding
- Stride

$$conv^*(im,f)_{x,y} = \sum_{i=1}^{N}\sum_{j=1}^{M} im\left(x - \frac{N-1}{2} + i - 1, y - \frac{M-1}{2} + j - 1\right) f(i,j) + b$$

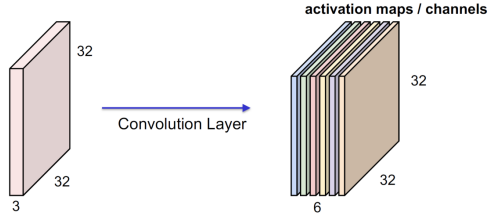
Weights of the filter that must be learned

Bias (to be learned)

25

Convolution Layer

If we had 6 5x5 filters, we'll get 6 separate activation maps.
These are also called "channels".

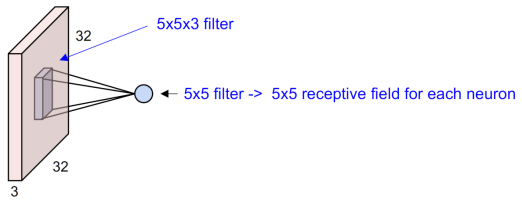


We stack these up to get a "new image" of size 32x32x6 !

26

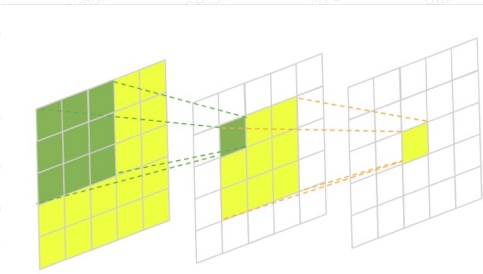
Convolution Layer Receptive field

- The spatial region in the input image that influences a neuron respond is called "receptive field".
- The deeper the layer is the larger the receptive field.



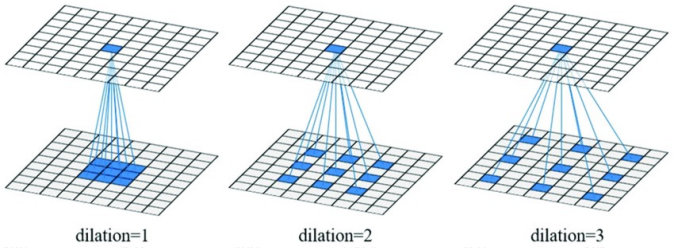
27

Convolution Layer: Receptive field



28

Convolution Layer: Receptive field



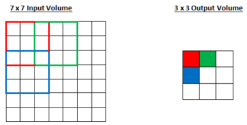
Effective receptive field 3x3 | Effective receptive field 5x5 | Effective receptive field 7x7

29

Convolution Layer: Stride

A closer look at spatial dimensions: Stride

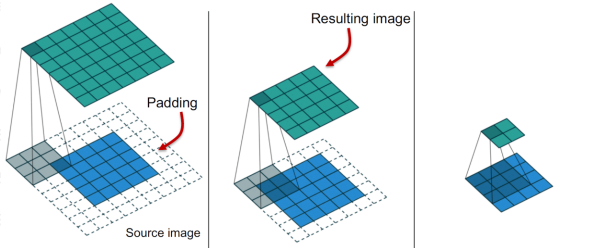
- Sometimes we want to scale down the resulting convolution (in the spatial domain).
- It can be implement by down-sampling the results.
- This operation is called "**Striding**".



30

Convolution Layer: padding

- Assume Image size $N \times N$, filter size $F \times F$:



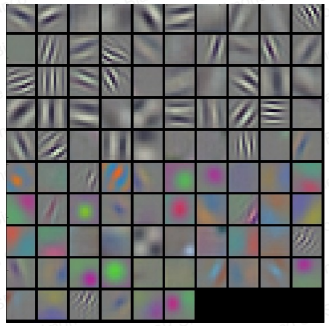
5x5 image, 3x3 filter	5x5 image, 3x3 filter	4x4 image, 3x3 filter
Full (full padding)	**Same (half padding)**	**Valid (no padding)**
Output = $(N + F - 1)$	Output = N	Output = $(N - F + 1)$
Padding = $(F - 1)$	Padding = $(F - 1)/2$	Padding = 0

31

Linear convolution – Filter bank example

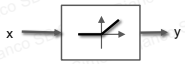
A bank of 96 filters of the first convolutional layer of AlexNet trained on ILSVRC2012 dataset

- Each one is a 11 × 11 pixels 3D filter (it applies to a RGB image)



32

Activation functions – Definition



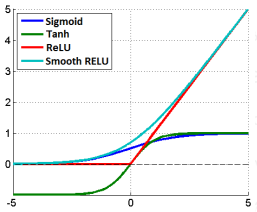
- Scalar non-linearity

$y = \frac{1}{1+e^{-x}}$ Sigmoid

$y = \tanh x$ Hyperbolic tangent

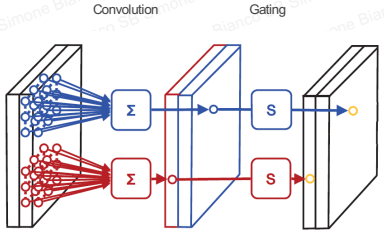
$y = \max\{0, x\}$ Rectified Linear Unit (ReLU)

$y = \log\ (1 + e^{x})$ Smooth ReLU



33

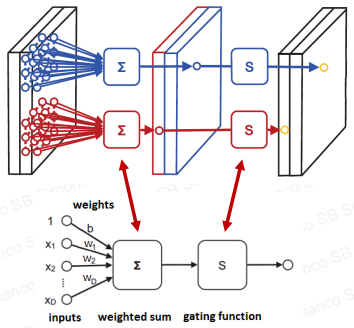
Activation functions – As neural network



Filters are followed by non-linear operators (e.g. gating function)

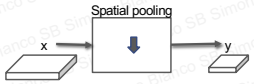
34

Activation functions – As neural network

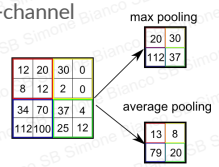


35

Spatial pooling



- Pooling and sub-sampling
- Encodes translation invariance
- Pooling computes the average/max of the features in a neighborhood
- It is applied channel-by-channel

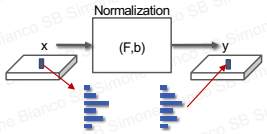


$y_{ijk} = \max_{pq \in \Omega_{ij}} x_{pqk}$ Max pooling

$y_{ijk} = \operatorname{avg}_{pq \in \Omega_{ij}} x_{pqk}$ Average pooling

36

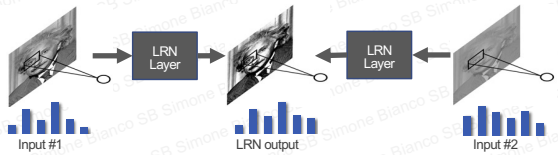
Local Response Normalization (LRN)



Contrast normalization

Effects:

- Improves invariance
- Improves optimization
- Improves sparsity

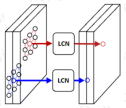


We want the same response

37

Local Response Normalization (LRN)

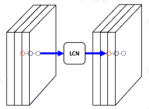
- WITHIN CHANNEL
  - Operates independently on different feature channels
  - Rescales each input feature basing on a local neighborhood



$$y_{ijk} = x_{ijk}\left(\kappa + \alpha \sum_{(u,v)\in N(i,j)} x_{uvk}^2\right)^{-\beta}$$

ACROSS CHANNELS

- Operates independently at each spatial location and groups of channels
- Normalizes groups G(k) of feature channels
- Groups are usually defined in a sliding window manner



$$y_{ijk} = x_{ijk}\left(\kappa + \alpha \sum_{q\in G(k)} x_{ijq}^2\right)^{-\beta}$$

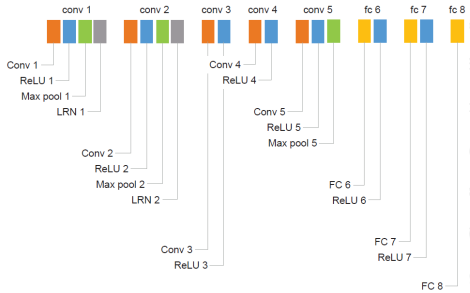
38

CNNs: summary

- CNNs are **biologically-inspired** architectures
- Two key differences to Standard Neural Nets: neurons arranged in **3D volumes** have **local connectivity**, **share weights**
- Differently from traditional Neural Networks, CNNs have other special layers
- Spatial pooling
- Local response normalization

39

CNNs: summary, typical architecture



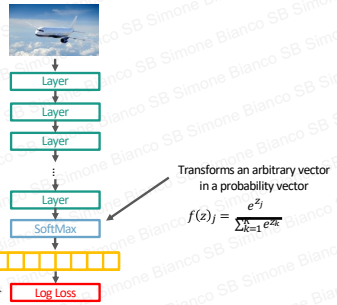
A. Krizhevsky et al., **Imagenet classification with deep convolutional neural networks**, *Advances in neural information processing systems, 2012.*

40

Convolutional Neural Networks (CNNs)

In order to optimize the weights in a neural network, we need to first define a loss function that we minimize over the training examples.

- General schema for classification



Transforms an arbitrary vector in a probability vector

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Drives the training of the classification task

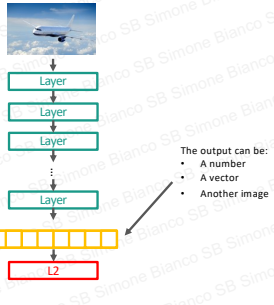
$$L = -\log(P(cur\_class))$$

41

Convolutional Neural Networks (CNNs)

In order to optimize the weights in a neural network, we need to first define a loss function that we minimize over the training examples.

- General schema for regression



The output can be:
- A number
- A vector
- Another image

Drives the training of the regression task

$$l2 = \sqrt{\sum_{i=1}^{N} (z_i - y_i)^2}$$

42

42

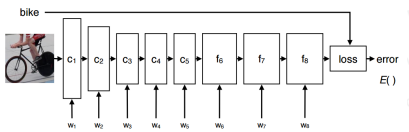
AA 2022/2023

CNN training

43

Training CNNs

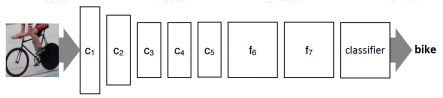
- Find CNN's weights **w** to minimize the **error** between ground truth labels and estimated labels
- Minimize using Stochastic Gradient Descent (SGD)
- This training method is called Backward-propagation



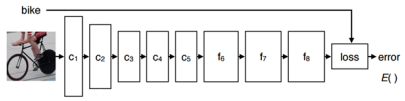
$$\text{argmin } E(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_8)$$

44

Training CNNs



- A **score function** maps the raw image pixels to class scores (e.g. a linear function)
- A **loss function** measures the **error** between ground truth labels and estimated labels on the training data (e.g. Softmax/Euclidean)



45

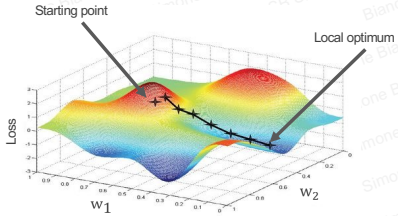
Training CNNs

- Challenge
  - Many parameters
  - Prone to overfitting
- Key ingredients
  - Very large datasets
  - Stochastic Gradient Descent (SGD)
  - Regularization methods
  - Graphic Processing Unit (GPU)

46

Stochastic Gradient Descent (SGD)

- Two steps
1. Forward-propagation
2. Backward-propagation



47

Stochastic Gradient Descent (SGD)

- Two steps
1. Forward-propagation
2. Backward-propagation
- Refinements
  - **Epochs:** all samples are visited sequentially, but in random order

48

Stochastic Gradient Descent (SGD)

- Two steps
1. Forward-propagation
2. Backward-propagation

- Refinements
  - **Epochs:** all samples are visited sequentially, but in random order
  - **Validation:** evaluate on an held-out validation set to diagnose convergence

49

Stochastic Gradient Descent (SGD)

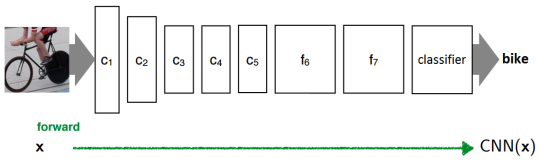
- Two steps
1. Forward-propagation
2. Backward-propagation

- Refinements
  - **Epochs:** all samples are visited sequentially, but in random order
  - **Validation:** evaluate on an held-out validation set to diagnose convergence
  - **Learning rate:** step size for each iteration
  - **Momentum:** previous weight updates are remembered at each iteration

50

Forward-propagation

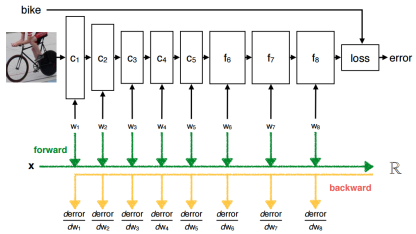
- Process of computing the output of the network given its input



51

Backward-propagation

- Process of computing gradients of the loss w.r.t. parameters in a multi-layer neural network
- Compute derivatives using **Chain Rule**



52